Electronic Version
Stylesheet Version v1.1.1

Description

METHOD AND APPARATUS FOR COMPUTERIZED EXTRACTING OF SCHEDULING INFORMATION FROM A NATURAL LANGUAGE E-MAIL

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The invention relates to computerized extracting of scheduling information from an e-mail, and more particularly, to a method and apparatus for computerized extracting of scheduling information from a natural language e-mail for automatic entry into an electronic calendar.

[0003] 2. Description of the Prior Art

[0004] Electronic devices are often used to help with personal organization. Calendars are often used to help us remember important dates in the future. Electronic calendar applications allow users to set events and alarms for particular

dates and times in the future and automatically remind the user of the event.

[0005] Electronic mail (e-mail) has also become an important part of todays modern society. People use e-mail to keep in touch with family and friends as well as to conduct business correspondence. E-mail often contains information about scheduling events that users will want to put in a personal calendar application. Examples of such events include: meetings, conferences, seminars, appointments, engagements, deadlines, etc. Fig.1 shows an example e-mail 100 containing scheduling information relating to a business presentation.

[0006] Upon reception of the e-mail 100, in order to remember the dates and times, the information listed for each event must be added to the calendar application. Because the e-mail 10 is a natural language (English) e-mail and does not follow a specific format recognized by the calendar application, the user must manually enter each event into the calendar. This is a time consuming and error prone operation. A need exists for an automatic agent that can help the user extract scheduling information from a natural language e-mail and directly export it to a calendar application.

[0007] In US Patent No. 6,035,278, Mansour describes a meeting scheduling tool provided for a user to search for an un-scheduled time block,setup a meeting for the time block, and manage the schedule arrangement. The meeting scheduling tool can be connected to an e-mail system, however, this is only for notification purposes of meetings configured using the scheduling tool and does not address the method of gathering scheduling information from an incoming e-mail.

[0008] In US Patent No. 6,094,681, Shaffer ,et al.describe an automated event notification apparatus including a data filter capableof analyzing data contained in e-mail messages, scheduling updates and requests transmitted to an electronic calendar of computer, and scheduling reminders transmitted by the electronic calendar.However, the automated event notification apparatus simply uses the data filter to determine whether an event has occurred or not. If the event has occurred, a user is notified using one of a plurality of notification methods. There is no provision for extracting information from scheduling updates other than to check for whether a certain event has occurred. The data filter makes a binary decision, and the event is determined to have either occurred or not oc-

curred.

[0009] In US Patent No. 6,272,532, Feinleib describes a centralized electronic reminder system for parsing received e-mail messagesto create reminder electronic messages. Natural language e-mails are received, parsed, and reminder information is extracted. When the date matches a date specified in the reminder information, the electronic reminder system constructs an e-message which is sent to a recipient specified in the reminder information. Although the electronic reminder system uses natural language e-mails, there is no ability to extract scheduling information from the e-mails to export to a personal calendar application. Additionally no detailed information is disclosed for a suitable method of parsing the e-mail to extract the desired scheduling information.

[0010] In published US patent application No. 20020174185, Jai, et al. describe a centralized system for capturing electronic data by parsing incoming e-mail header and data content and, depending on the type of the e-mail, selectively extracting data from the e-mail and forwardingthe extracted data according to user preferences. Although the system can use various formats and types of e-mails, there is no ability for a user to extract scheduling infor-

mation from the e-mails to automatically export to a personal calendar application. Additionally, there is again no detailed information disclosed for a suitable method of parsing the e-mail to extract desired scheduling information.

[0011] The patents listed above do not help a user extract scheduling information and input the scheduling information to a calendar application located at the users location. Furthermore, there is a need for a suitable parsing method for natural language e-mails to allow scheduling information contained in the e-mail to be properly extracted.

SUMMARY OF INVENTION

[0012] It is therefore a primary objective of the claimed invention to provide a method and apparatus for computerized extracting of scheduling information from a natural language e-mail for automatic entry into a calendar application, to solve the above-mentioned problems.

[0013] According to the claimed invention, a method for computerized extracting of scheduling information from a natural language text for automatic entry into a calendar application is disclosed. The method comprises the following steps: (a) Parse the natural language text to build a de-

pendency tree. (b) Determine if the natural language text contains scheduling information by calculating a probability sum for the dependency tree. (c) If the probability sum exceeds a predetermined value, extract scheduling information from the dependency tree and export the scheduling information to the calendar application.

[0014] According to the claimed invention, a personal organization apparatus comprising a processor for executing code in the personal organization apparatus, and a storage unit connected to the processor for storing data used by the processor including a natural language text. The processor parses the natural language text to build a dependency tree in the storage unit, determines if the natural language text contains scheduling information by calculating a probability sum for the dependency tree, and if the probability sum exceeds a predetermined value, extracts scheduling information from the dependency tree and exports the scheduling information to a calendar application.

[0015] These and other objectives of the claimed invention will no doubt become obvious to those of ordinary skill in the art after reading the following detailed description of the preferred embodiment that is illustrated in the various

figures and drawings.

Brief Description of Drawings

[0016] Fig.1 is an example natural language e-mail containing scheduling information according to the prior art.

[0017] Fig.2 is a dependency pair diagram for a sample sentence according to the present invention.

[0018] Fig.3 is a flowchart describing extracting scheduling information from an incoming e-mail according to the present invention.

[0019] Fig.4 is a flowchart describing how to build the dependency database used by the parsing step in Fig.4.

[0020] Fig.5 is a flowchart describing the parsing step of Fig.3.

[0021] Fig.6 is a dependency tree diagram for the e-mail of Fig.1 according to the present invention.

[0022] Fig.7 is a flowchart describing the determining if the e-mail contains scheduling information step of Fig.3.

[0023] Fig.8 is a flowchart describing the extract scheduling information step of Fig.3.

[0024] Fig.9 is a block diagram of a personal organization apparatus for implementing the flowchart of Fig.3.

Detailed Description

[0025] Natural human languages are both varied and compli-

cated, however, in any text understandable by a human, all elements of the text are related in a certain way. In order to extract scheduling information from a natural language text, it is necessary to determine candidates for the scheduling information and decide whether these candidates contain scheduling information. The assumption is that information in a natural language appears in a specific pattern for specific types of information. According to the present invention, dependency grammar is used to define a base set of possible pattern structures. Dependency grammar defines this basic structure as word pairs (dependency pairs) and constructs larger patterns (dependency trees) using these dependency pairs. Incoming E-mail messages are parsed to gather dependency pairs applicable to scheduling information and the result is a possible scheduling event.

[0026] Fig.2 shows a dependency pair diagram 200 for a sample sentence according to the present invention. According to dependency grammar, each word in a sentence has one and only one dominating head word. This forms a basic set of dependency pairs for each sentence as indicated with arrows, the arrow pointing from the head word to a dependent word. In Fig.2, the word "meet" is the root of

the sentence and is the head to "shall", "we", "each-other", "at", "tomorrow" and "2:00PM". If "meet" can be identified as the root of the sentence, a dependency tree can be built for the sentence showing all the decedents (tails) of the root and subsequent dependency pairs.

[0027] Fig.3 shows a flowchart 300 describing using dependency grammar rules to extract scheduling information according to the present invention. The flowchart contains the following steps:

[0028] Step 302:Wait for an incoming e-mail to arrive. When a new e-mail arrives proceed to step 304.

[0029] Step 304:Parse the incoming e-mail to build a dependency tree and proceed to step 306.

[0030] Step 306:Determine if the e-mail contains scheduling information by calculating a probability sum for the dependency tree. Each dependency pair in the tree has an associated probability, which represents how often this particular word pair appears in natural language e-mails concerning scheduling information. If the probability sum exceeds a predetermined value then assume the e-mail contains scheduling information and proceed to step 308, otherwise return to step 302 to wait for the next incoming e-mail.

[0031] Step 308:Extract the scheduling information from the dependency tree, export the scheduling information to a calendar application, and when finished, return to step 302 to wait for the next incoming e-mail.

[0032] During the parsing step 302, the dependency tree does not need to contain every dependency pair from the e-mail but only the pairs relevant to scheduling information. This requires a decision to be made of whether each word pair is relevant to scheduling information. Additionally, for a given word pair, a decision must be made to determine which word is the most likely head word. In order to make these decisions, a dependency database is used as a reference. The dependency database contains a list of dependency pairs that are frequently found in a natural language e-mail corpus, and the e-mail corpus contains many e-mails containing scheduling information. Each dependency pair in the dependency database has an associatedprobability that corresponds to how often the particular word pair is found in the e-mail corpus. It should be noted that, in the preferred embodiment of the present invention, this probability is stored as a log-probability to facilitate adding the probabilities of different dependency pairs together.

[0033] Fig.4 shows a flowchart 400 describing how to build the dependency database using an e-mail corpus, an associated tagged corpus, a head word list, and a set of violation rules. The e-mail corpus contains a plurality of sample natural language e-mails containing scheduling information. The tagged corpus specifies actual head words for sentences relevant to scheduling information in the e-mail corpus and contains dependencies for all other words in each sentence with respect to the head words. The head word list contains possible head words, and the violation constraints specifyillegal dependency structures. Examples of possible illegal dependency structures include pronouns acting as head words to verbs, or adjectives acting as head words to nouns. The e-mail corpus, tagged corpus, head word list, and violation constraints are specified in advance and provided by the system designer. Building the dependency database is described in flowchart 400 and comprises the following steps:

[0034] Step 402:Start at the first sentence. Examine each sentence one by one starting with the first sentence. Proceed to step 404.

[0035] Step 404:Segment the sentence into words. Some languages use a base element other than words. For exam-

ple, the Chinese language uses characters, and these characters must be grouped into words having meaning. This grouping is referred to as segmenting. Additionally, even in English, some words need to be paired with other words to form a proper meaning. For example, in Fig.2, the word "each-other" is actually the words "each" and "other" grouped together to form a single word. As segmenting is well known to people skilled in the art, further description is hereby omitted. If segmenting is not required, step 404 can be removed from the flowchart 400, otherwise proceed to step 406 when finished.

[0036] Step 406:Build word pairs for the sentence. Make a list of all combinations of words in the sentence. Proceed to step 408.

[0037] Step 408:Start at the first word pair. Examine each word pair one by one starting with the first word pair. Proceed to step 410.

[0038] Step 410:Does the word pair occur frequently in the e-mail corpus? If yes then proceed to step 412, otherwise skip to step 418.

[0039] Step 412:Determine the head word. Check the head word list and the tagged corpus to determine which word in the word pair should be designated the head word. Proceed to

step 414.

[0040] Step 414:Is the resulting word pair structure a valid dependency pair? Use the violation rules to determine if the syntax of the word pair is allowable. Additionally check the dependency database to ensure the word pair is not already included in the dependency database and to determine if this word pair structure should be added as a new dependency pair, e.g. this word pair fits with an existing dependency tree in the dependency database. If the word pair is a new and valid dependency pair then proceed to step 416, otherwise skip to step 418.

[0041] Step 416:Add the word pair as a dependency pair to the dependency database and proceed to step 418.

[0042] Step 418:Are the word pairs finished? If yes then proceed to step 422, otherwise proceed to step 420.

[0043] Step 420:Examine the next word pair and proceed to step 410.

[0044] Step 422:Are the sentences finished? If yes then proceed to step 426, otherwise proceed to step 424.

[0045] Step 424:Examine the next sentence and proceed to step 404.

[0046] Step 426:Was at least one new dependency pair added to the dependency database? If no then the dependency

database is finished being built, otherwise repeat the procedure to look for next level dependency pairs by returning to step 402.

[0047] Fig.5 is a flowchart 500 describing the parsing step 304 of Fig.3. Once the dependency database is built using the flowchart 400 shown in Fig.4, the parsing step 304 of Fig.3 can be broken down into the following steps:

[0048] Step 502:Start at the first sentence. Examine each sentence one by one starting with the first sentence. Proceed to step 504.

[0049] Step 504:Segment the sentence into words. This step is the same as step 404 in Fig.4 and can similarly be omitted if not required. Proceed to step 506 when finished.

[0050] Step 506:Build a head word list. Use the dependency database to make a list of all words in the sentence that are possible head words. Proceed to step 508.

[0051] Step 508:Start at the first possible head word. Examine each head word one by one starting with the first possible head word. Proceed to step 510.

[0052] Step 510:Build word pairs for the sentence. Make a list of all two-word combinations of words in the sentence. Proceed to step 512.

[0053] Step 512:Start at the first word pair. Examine each word

pair one by one starting with the first word pair. Proceed to step 514.

[0054] Step 514:Is the word pair in the dependency database? If yes then proceed to step 516, otherwise skip to step 518.

[0055] Step 516:Add the word pair as a dependency pair to the dependency tree. Proceed to step 518.

[0056] Step 518:Are the word pairs finished? If yes then proceed to step 520, otherwise proceed to step 524.

[0057] Step 520:Examine the next word pair, and proceed to step 514.

[0058] Step 522:Are the possible head words finished? If yes then proceed to step 526, otherwise proceed to step 524.

[0059] Step 524:Examine the next possible head word and proceed to step 508.

[0060] Step 526:Are the sentences finished? If yes then the parsing procedure is finished so proceed to step 306 in Fig.3, if not then proceed to step 528.

[0061] Step 528:Examine the next sentence and proceed to step 504.

[0062] Fig.6 shows a dependency tree 600 for the e-mail 100 of Fig.1 resulting from the parsing steps described in Fig.5. Because there are multiple scheduling events contained in the same e-mail 100, there are actually five sub-trees

602, 604, 606, 608, and 610 listed in the dependency tree 600, and each sub-tree contains only the dependency pairs that relate to scheduling information as determined from the dependency database. The first sub-tree 602 is based on the root word "reserved", the second sub-tree 604 is based on the root word "finish", the third sub-tree 606 is based on the root word "review", the fourth sub-tree 608 is based on the root word "rehearsal", and the fifth sub-tree 610 is based on the root word "meet".

[0063] Fig.7 shows a flowchart 700 describing the determining if the e-mail contains scheduling information step 306 of Fig.3 and contains the following steps:

[0064] Step 702:Sum the log-probability of the dependency tree and proceed to step 704.

[0065] Step 704:Is the log-probability sum greater than a predetermined value? If yes then assume the e-mail contains scheduling information and proceed to step 308 for extraction. If the log-probability sum is not greater than the predetermined value then assume the e-mail is unrelated to scheduling information and return to step 302 to wait for the next incoming e-mail.

[0066] Fig.8 shows a flowchart 800 describing the extract scheduling information step 308 of Fig.3 and contains the

following steps:

[0067] Step 802:Start at the first dependency sub-tree in the dependency tree. Each sub-tree in the dependency tree contains scheduling information for a particular event so extract the scheduling information one by one starting with the first sub-tree. Proceed to step 804.

[0068] Step 804:Start at the first dependency pair. Examine each dependency pair one by one starting with the first dependency pair. Proceed to step 806.

[0069] Step 806:Does the category match a predetermined scheduling type? If yes then proceed to step 808, otherwise skip to step 812.

[0070] Step 808:Compute a scheduling value according to the scheduling type and proceed to step 810.

[0071] Step 810: Register the scheduling value with the application specific format required by the calendar application in an add list and proceed to step 812.

[0072] Step 812:Are the word pairs finished? If yes then proceed to step 816, otherwise proceed to step 814.

[0073] Step 814:Examine the next word pair and proceed to step 806.

[0074] Step 816:Are the dependency sub-trees finished? If yes then proceed to step 820, otherwise proceed to step 818.

[0075] Step 818:Examine the next dependency sub-tree and proceed to step 804.

[0076] Step 820:Confirm the add list with the user. The user may want to modify the scheduling information in the add list before it is export to the calendar application. When the information in the add list is confirmed by the user, proceed to step 822.

[0077] Step 822:Export the schedulinginformation contained in the add list to the calendar application. When finished, return to step 302 to wait for the next incoming e-mail.

[0078] Fig.9 shows a block diagram of a personal organization apparatus 900 for implementing the flowchart 300 of Fig.3. A personal organization apparatus includes electronic devices that assist with scheduling and can include such devices as desktop computers, laptops computers, notebooks, personal digital assistants (PDAs), and cell phones. The personal organization apparatus 900 includes a processor 902 and a storage unit 904 connected to the processor. The storage unit 904 provides storage capability for an incoming e-mail 906, a dependency database 910, and code for a calendar application 914. Optionally, a natural language e-mail corpus 912 can also be stored in the storage unit to allow the personal organi-

zation apparatus 900 to build the dependency database 910. Additionally, a user interface 916 and a network interface 918 are connected to the processor for interfacing with a user and for receiving the incoming e-mail 906 respectively. The storage unit 904 can be implemented with a memory device such as a RAM, a storage device such as magnetic or optical media, or a combination of different types of memory and storage devices.

[0079] A natural language e-mail 906 is received from the network interface 918 and stored in the storage unit 904 by the processor 902. The e-mail 906 is parsed by the processor 902 to build the dependency tree 920. The dependency tree 920 contains word pairs from the e-mail 906 that are found in the dependency database 910, and the word pairs are stored as dependency pairs in a tree structure in the dependency tree 920. Each sub-tree in the dependency tree 920 deals with word pairs that are dependent on a common root head word. For most scheduling e-mails, this means each sub-tree contains information particular to a particular scheduling event.

[0080] When all the word pairs in the e-mail 906 have been checked, the processor 902 sums a log probability of the dependency pairs in the dependency tree 920. Each de-

pendency pair has an associated probability value also stored (preferable in log probability form) in the dependency tree, the probability value corresponding to how frequently this particular word pair was found in the e-mail corpus 912. More frequently appearing word pairs are given higher probability of indicating whether or not the e-mail 906 contains scheduling information. If the log probability sum is higher than a predetermined value, the e-mail 906 is assumed to contain scheduling information. In this case the processor 902 extracts the scheduling information by recursively checking each dependency pair in each sub-tree of the dependency tree 920. For each dependency pair, the processor 902 checks to see if the dependency pair matches a scheduling category that needs conversion to a scheduling value. Examples of such values are words like "tomorrow", or "next week". Depending on the date of the e-mail, these words need to be converted to an actual date. Next, the processor 902 registers the value of the dependency pair with the format specific requirements for the calendar application 914. For example, if the dependency pairs are stored in the form (head, tail), given required calendar application 914 parameters of PERSON, PLACE, TIME, SUBJECT, the dependency pairs of

Fig.6 could be grouped and processed as follows:

[0081] (reserved, schedule)

[0082] (reserved, room) -> PLACE = room

[0083] (room,video-conference) -> PLACE = video-conference room

[0084]

[0085] (finish, 6/18) -> TIME = 6/18

[0086] (finish, material) -> SUBJECT = finish material

[0087]

[0088] (review, 6/23 15:00-1700) -> TIME = 6/23 15:00-1700

[0089] (review, material) -> SUBJECT = review material

[0090]

[0091] (rehearsal, 06/25 10:00-12:00) -> TIME = 6/25 10:00-12:00

[0092] (rehearsal, the) -> SUBJECT = the rehearsal

[0093] (rehearsal, first) -> SUBJECT = the first rehearsal

[0094]

[0095] (meet, tomorrow) -> TIME = 06/14

[0096] (meet, go-over) -> SUBJECT = go-over

[0097] (go-over, material) -> SUBJECT = go-over material

[0098] (meet, in-person) -> PERSON = Nancy

[0099]

[0100] When each dependency pair for each sub-tree in the dependency tree 920 has been registered in an add list, the list can be confirmed with the user using the user interface 916. The user is presented with the prospective add list and given an opportunity to modify the scheduling information contained in the add list. For example, the user may wish to change the time for a meeting to an earlier time to prevent tardiness. Once confirmed by the user, the processor 902 exports the add list to the calendar application 914 using the proprietary format of the particular calendar application 914. The processor 902 may also be required to confirm the calendar application 914 is ready to receive the new scheduling information and may need to send a confirmation message to the calendar application 914. Additionally, it should be noted that using dependency grammar rules, the processor 902 could build the dependency database 910 using the plurality of natural language e-mails containing scheduling information

stored in the e-mail corpus 912.

[0101] In contrast to the prior art, the method and apparatus for computerized extracting of scheduling information according to the present invention can extract scheduling information from a natural language e-mail and export the extracted scheduling information to a calendar application. Dependency grammar rules are used to specifically identify and connect the words of each sentence having to do with scheduling information. The scheduling information is extracted by recursivelyexamining word pairs having to do with the scheduling information contained in the e-mail. Additionally, although the previous detailed description deals specifically with natural language e-mails, the same method and apparatus could be directly used for any natural language texts.

[0102] Those skilled in the art will readily observe that numerous modifications and alterations of the device may be made while retaining the teachings of the invention. Accordingly, that above disclosure should be construed as limited only by the metes and bounds of the appended claims.